# IBM AI Hardware Accelerator Kit (AIHWKit)


**Students:** Gyujun Jeong, Rayyan Shah
**Mentors:** Kaoutar El Maghraoui, Andrea Fasoli, Hadjer Benmeziane

Georgia Tech Open Source Program Office

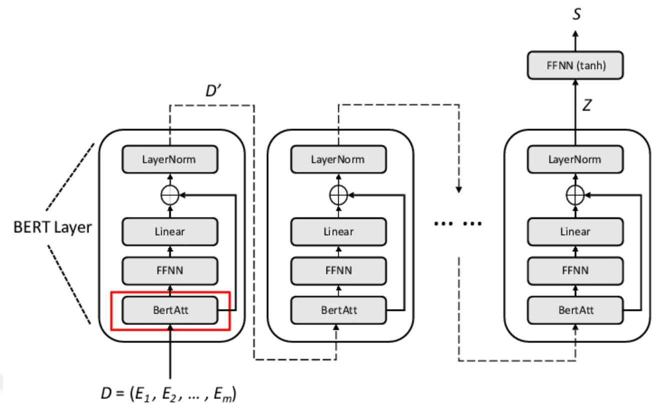

## Implementation and Noise Analysis for Generative Models (Gyujun Jeong)

**I. BERT Noise Analysis**
- **Objective**: Analyze the impact of noise on different layers of the BERT model using AIHWkit modules.

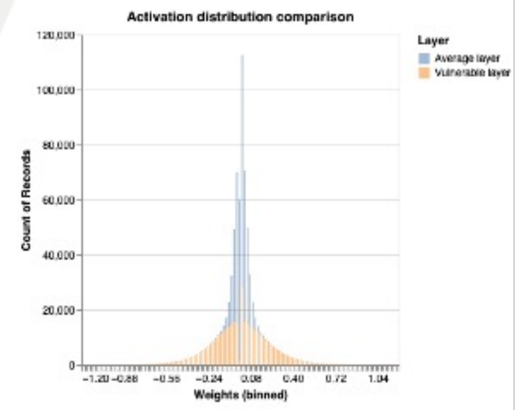
**II. GPT-2 Implementation**
- **Objective**: Implement GPT-2 using AIHWkit and evaluate its performance on AIHWkit.

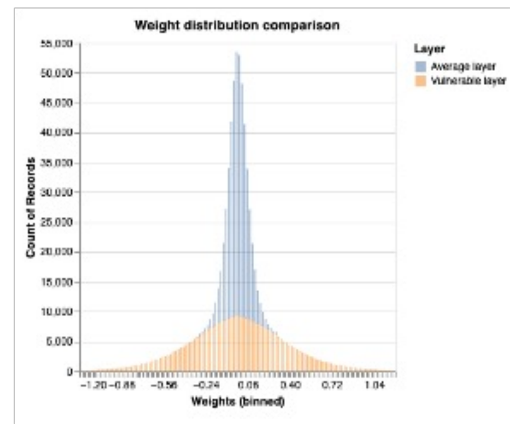
## I. BERT Noise Analysis [link]

**1. Project Description**
- **Key Findings**: Initial and final layers are the most vulnerable to noise
- Susceptible layers have wider distribution
- White Noise applied with std=1. F1 and EM scores evaluated for each noise-applied layer.



**2. Result**
- Initial Layers (Embeddings) and final layers (output dense) are vulnerable to noise
- As they handle the first conversion of raw input data and final prediction generation
- Layers performing complex transformations (e.g., dense layers with multiple weights) are more susceptible to noise
- The layers vulnerable to noise exhibit a wider spread in their distribution





**3. Outcome**
- Detailed noise impact analysis reveals crucial insights into model robustness.

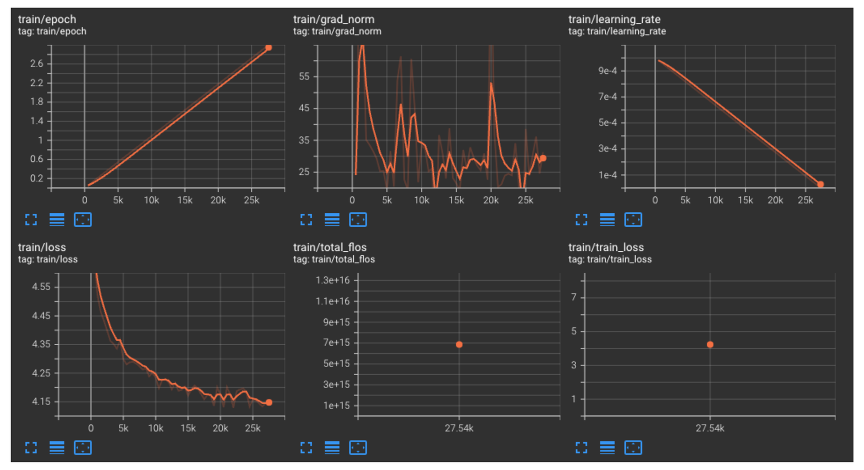
**4. Future Work**
- Investigate different types of noise and their propagation effects.

## II. GPT-2 Implementation [link]

**1. Project Description**
- **Key Findings**: GPT-2 implementation based text generation with AIHWKit
- Smallest model and datasets for demo
- Metrics: Validation Cross-Entropy & PPL.

```
# GPT-2 model from Hugging Face model hub
MODEL_NAME = "distilbert/distilgpt2" # Smallest GPT-2 model
TOKENIZER = AutoTokenizer.from_pretrained(MODEL_NAME)
```

**2. Result**
- Achieved coherent text generation.
- Validation Cross-Entropy (Loss):
- HWA = 4.059, PPL = 57.9;
- Digital = 3.3259, PPL = 27.8.



Input Prompt: `"Once upon a time"`

Output Prompt: `Once upon a time of war, the German army was able to defeat the German army in the Battle of the Rhine`

**3. Outcome**
Complex models can be effectively implemented using AIHWkit.

**4. Future Work**
- Optimize model parameters for better performance

## Analog AI Neural Architecture Search (Analog-NAS) – Rayyan Shah

**I. Creating and Integrating Additional Surrogate Models**
- **Objective**: leverage a surrogate model to predict the ranking of neural architectures within the search space.
- **Tasks:** MobileBERT Implementation for Question Answering

## I. Surrogate Model Training and Integration (MobileBERT) [link]

**1. Project Description**
- **Key Findings:** Integration of Surrogate models can enhance the prediction accuracy and efficiency of Analog-NAS.
- Surrogate models are trained on diverse datasets for tasks like question answering.
- **Training Datasets:** MobileBERT on SQuAD dataset for question-answering tasks.

**2. Result**
- **Training Process:** Used AIHWkit for hardware-aware training, leveraging the capabilities of in-memory computing devices.
- W&B (Weights & Biases) utilized for hyper-parameter optimization and logging.
- **Installation:** Installed AIHWkit and other necessary libraries such as wandb, accelerate and transformers.
- **Resistive Processing Unit Configuration:** Defined a StandardHWATrainingPreset RPU configuration for hardware-aware training.

```
from aihwkit.simulator.presets.inference import StandardHWATrainingPreset

# Define RPU configuration
rpu_config = StandardHWATrainingPreset()
```

- **Hyper-parameter Optimization:** Configured optimization using Bayesian methods, focusing on minimizing training loss with parameters such as batch size, learning rate, and weight decay.

```
# Setup optimizer
optimizer = AnalogAdam(model.parameters(), lr=learning_rate)

# Training loop
for epoch in range(num_training_epochs):
    model.train()
    for batch in train_dataloader:
        outputs = model(**batch)
        loss = outputs[0]
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
```

**3. Outcome**
- **Noise Resilience:** The model demonstrates robustness against hardware-induced noise, and drift at 1-Day Accuracy and Accuracy Variation over one Month (AVM). This resilience ensures consistent and reliable model outputs.
- **Energy Efficiency:** Utilizing analog layers can significantly lowers the power consumption of the MobileBERT model, making it ideal for deployment in edge computing scenarios where power efficiency is crucial.

**4. Future Work**
- Further refine the integration of MobileBERT by tuning hyper-parameters and experimenting with different training configurations.
- Explore additional datasets to expand the application scope of AnalogNAS. Potential tasks include lane segmentation and node classification.

Georgia Tech